# Queue Management for SLO-Oriented Large Language Model Serving

## 论文信息

**Title**: Queue Management for SLO-Oriented Large Language Model Serving

**Authors**: Archit Patke, Dhemath Reddy, Haoran Qiu, and Zbigniew Kalbarczyk, Ravishankar Iyer, University of Illinois Urbana-Champaign;

Saurabh Jha, Christian Pinto, and Chandra Narayanaswami, IBM Research;

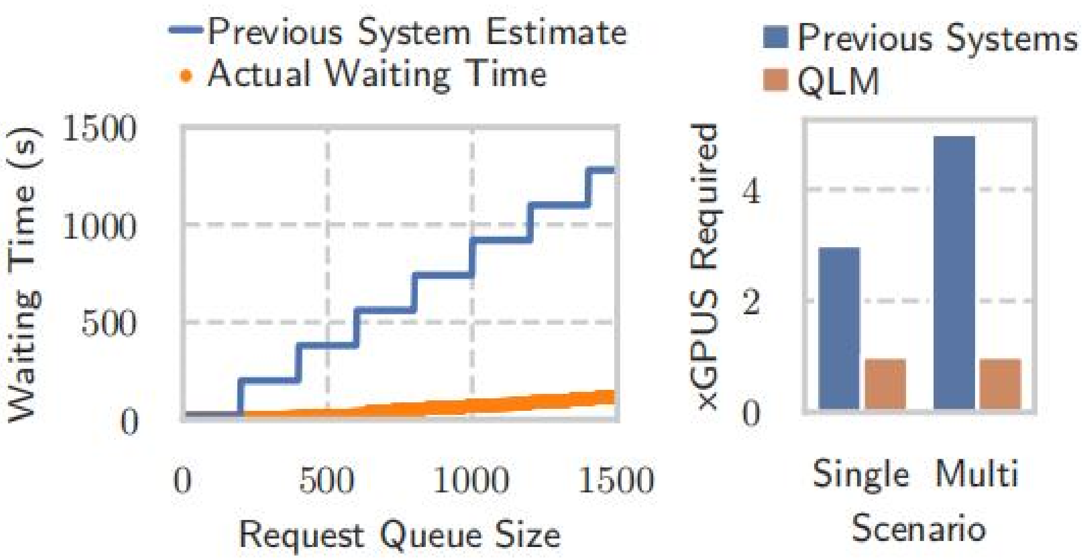
**Conference**: *SoCC '24, November 20–22,2024, Redmond, WA, USA*

## 研究背景

1. 为企业和消费者应用提供具有延迟导向的服务等级目标（SLO）的多个模型变得越来越关键。

2. 以往该领域的工作主要侧重于服务交互式请求，而未考虑批量请求。

3. 以往面向 SLO 的服务工作主要聚焦于具有确定性执行时间的传统深度神经网络（DNN）服务负载。




## 动机

见解#1：长请求队列中的等待时间可以通过分析进行准确估计。

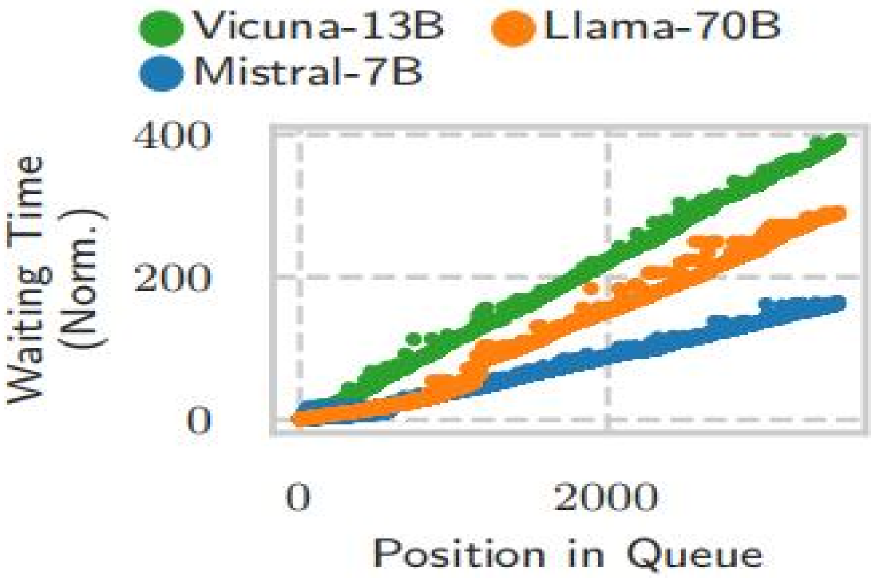



**Figure 3: Requests have predictable waiting times in a continuous batching system.**

见解#2：由于连续批处理造成的队头（HOL）阻塞时间可能长达数十秒。


**Figure 4: Forced request eviction leads to reduction in head-of-line (HOL) blocking time.**

见解#3：诸如最早截止期限优先（EDF）之类的策略不足以消除模型切换中的队头（HOL）阻塞。

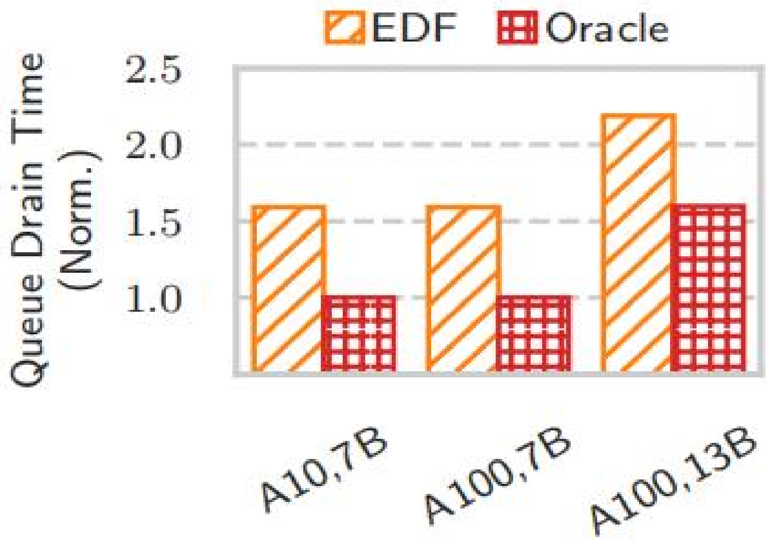



**Figure 5: Model swapping and request pulling can jointly decrease queue drain time.**

QLM 设计概览

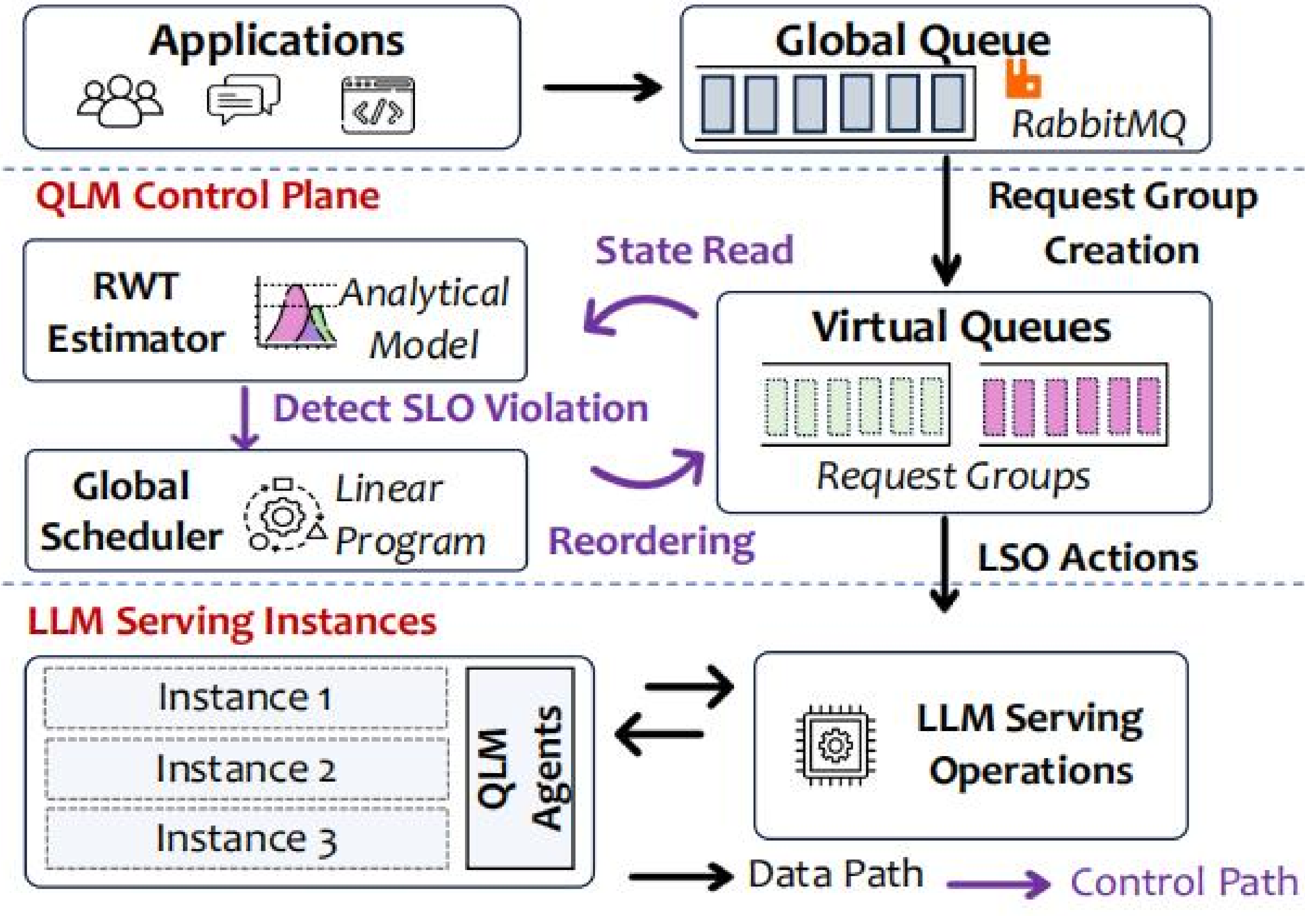



Figure 6: Overview of QLM.

## 请求组创建

1.当新请求加入全局队列时，它们被分类到现有的请求组中。

2.触发请求等待时间（RWT）估计器计算，以检查是否有任何 SLO 被违反。

3.一旦发现任何 SLO 违反，将调用全局调度器来重新排列虚拟队列中的请求组，以最大化 SLO 达成率。

（1）基于模型类型、输入/输出 token 分布和 SLO 值对类似请求进行聚类。

（2）拆分大型请求组。

**Algorithm 1** Request Group Creation

```
groups ← kMeansClustering(requests)
for i ← 1 to length(groups) do
    if groups[i].size() > avg_batch_size × δ then
        newGroups ← groups[i].splitHalf()
        groups.append(newGroups)
    end if
end for
```

## LLM 服务操作

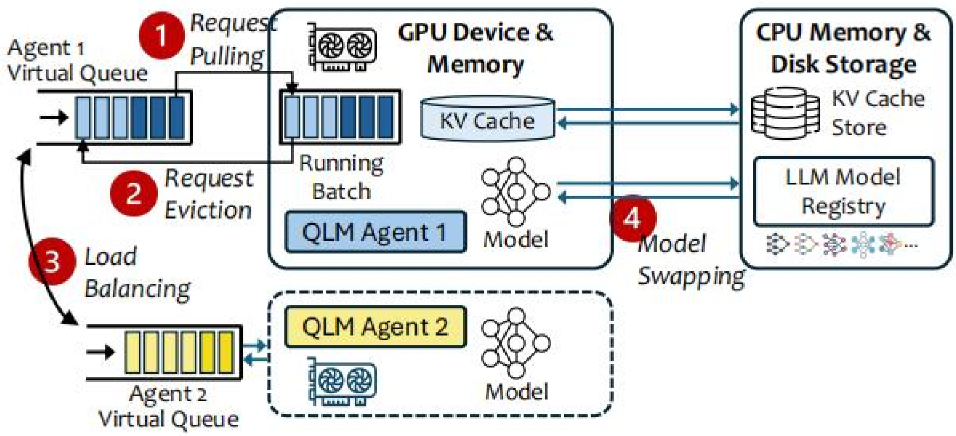



**Figure 7: Basic LLM serving operations (LSOs) for an LLM-serving instance that a QLM agent manages.**

## 请求等待时间估计器

The total request completion time equals the sum of the waiting time (Wq), prefill time (P), and total decode time across all the output tokens (Dq) for a request q.

$$C_q = W_q + P + D_q \quad (1)$$

$$W_q = \sum_{i=1}^{q-1} \frac{O_i}{\Theta} \tag{2}$$

$$\sum_{i=1}^{q-1} O_i \sim N((q-1)\mu_o, (q-1)\sigma_o^2) \tag{3}$$

$$D_q = O_q \times \epsilon \times d \tag{4}$$

$$C = \max_q C_q \tag{5}$$

全局调度器

$$\sum_g \sum_j x_{g,i,j} = 1 \forall i \quad \sum_i x_{g,i,j} = 1 \forall g, j \tag{6}$$

$$m_{g,j} = \sum_i \mathrm{models}_i \times x_{g,i,j} \forall g, j \tag{7}$$

$$slo_{g,j} = \sum_i \mathrm{slos}_i \times x_{g,i,j} \forall g, j \tag{8}$$

$$t_{g,j} = (m_{g,j-1} \neq m_{g,j}) \forall g, j \tag{9}$$

$$\mathrm{wt}_{g,j} = \sum_i \sum_k^{j-1} W_{g,i} \times x_{g,i,k} + \sum_k^{j-1} t_{g,k} \times S + \sum_i \sum_k^{j-1} C_{g,i} \times t_{g,k} \times x_{g,i,j} \forall g, j \tag{10}$$

$$p_{g,j} = wt_{g,j} - slo_{g,j} \forall g, j \qquad (11)$$

$$p_{g,j} \leq 0 \forall g, j \qquad (12)$$

$$\min(\sum_{g}\sum_{j} p_{g,j}) \qquad (13)$$

## 实验设置

我们在多个不同规模的开源 LLM 上评估 QLM：Mistral-7B、Vicuna-13B 和 Llama-70B。

基线：EDF（最早截止期限优先）、vLLM 和 SHEPHERD。

测试平台：我们在由两种类型的 GPU 组成的测试平台上进行评估：30 个 NVIDIA A10（24GB 内存）和 50 个 NVIDIA A100（80GB 内存）。

工作负载：请求到达遵循泊松分布，并通过改变到达率来创建队列。每个工作负载跟踪使用来自 ShareGPT 数据集的 3500 个请求。我们将所有请求分为三类，并相应地定义它们的 SLO 值：（1）交互式：20 秒，（2）批量 1：1 分钟，（3）批量 2：1 小时。

WA：单模型交互式和批量工作负载

WB：多模型批量工作负载

WC：单模型大型提示工作负载

## 单模型评估

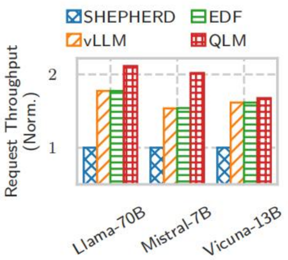



Figure 9: Single model request serving throughput at 0.5K requests/s interactive arrival rate. Increased throughput corresponds to 1.1-2.3× GPU requirement reduction.

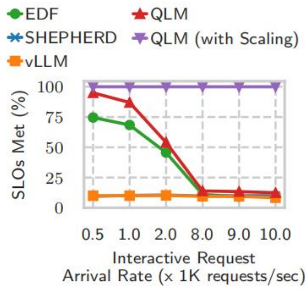



Figure 10: Single model SLO satisfaction for varying interactive request arrival rates for Vicuna 13B.

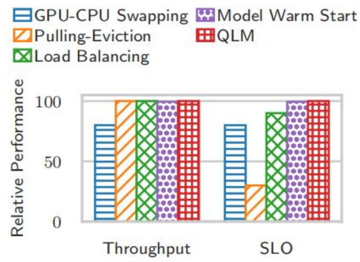



Figure 11: Single model LSO ablation study at 0.5K requests/s interactive arrival rate for Vicuna 13B.

# 多模型评估

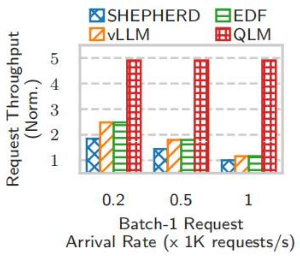



Figure 12: Multi-model request serving throughput for varying Batch-1 request arrival rates. Increased throughput corresponds to 2-5× GPU requirement reduction.

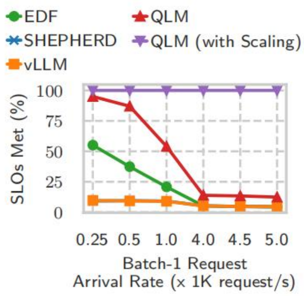



Figure 13: Multi-model SLO satisfaction for varying Batch-1 request arrival rates.

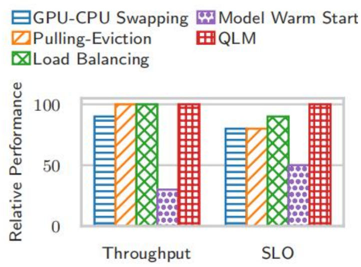



Figure 14: Multi-model LSO ablation study for 0.25K requests/sec Batch-1 arrival rate.

# QLM 鲁棒性分析

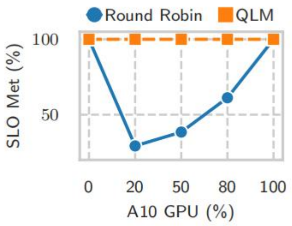



Figure 15: Impact of hardware heterogeneity.

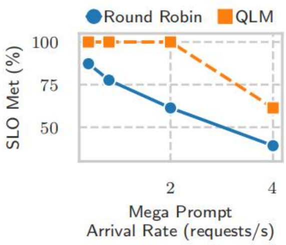



Figure 16: Impact of mega prompt arrivals.

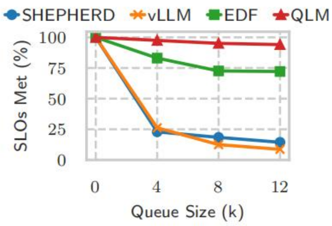



Figure 17: Impact of increasing queue size on SLO satisfaction.

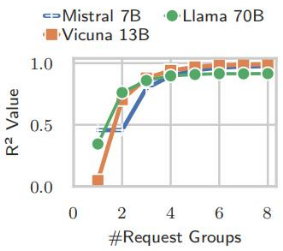



Figure 18: Accuracy of RWT estimator.

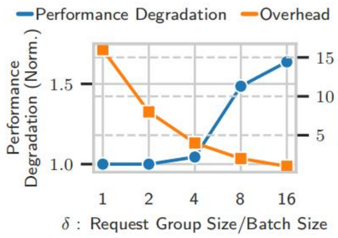



Figure 19: Impact of request group size on QLM performance.


Figure 20: QLM Overhead.

# 结论

1.我们提出了 QLM,这是一种新颖的队列管理框架,用于面向服务等级目标(SLO)的大型语言模型(LLM)服务后端协调。

2.在异构模型类型和 GPU 设备上使用真实世界 LLM 服务数据集进行的评估表明,QLM 将端到端延迟 SLO 达成率提高了 40–90%,同时使服务吞吐量和设备利用率提高了 20-400%。